(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 0 973 294 A2

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
19.01.2000 Bulletin 2000/03

(51) Int. Cl.[7]: **H04L 12/18**, H04L 29/06

(21) Application number: **99305083.0**

(22) Date of filing: **28.06.1999**

(84) Designated Contracting States:
AT BE CH CY DE DK ES FI FR GB GR IE IT LI LU MC NL PT SE
Designated Extension States:
AL LT LV MK RO SI

(30) Priority: **29.06.1998 US 106403**

(71) Applicant: **MICROSOFT CORPORATION**
Redmond, Washington 98052-6399 (US)

(72) Inventors:
- **Hamilton, Keith S.**
  Redmond, WA 98053 (US)
- **Meizlik, Robert Steven**
  Newcastle, WA 98059 (US)

(74) Representative:
Orr, William McLean
Urquhart-Dykes & Lord,
Tower House,
Merrion Way
Leeds LS2 8PA (GB)

(54) Method and computer program product for efficiently and reliably sending small data messages from a sending system to a large number of receiving systems

(57) In a network with a sending system networked to at least one receiving system, it is sometimes desirable to transfer relatively short messages between the sending system and one or more receiving systems in a highly reliable yet highly efficient manner. The present invention defines two short message protocols, one of which relies on a statistical model and the other of which uses positive acknowledgement to track receipt of transmitted packets by intended recipient. The statistical reliability mode is based on the observation that for each packet in a message that is transmitted, the probability that at least one packet of the message is received by a given system increases. Thus, in the statistical reliability mode messages are divided into a guaranteed minimum number of packets, with additional packets being added if the message length is insufficient to fill the minimum number of packets. The positive reliability mode of the present invention periodically sets an acknowledgement flag in the packets transmitted for a message. Receiving systems send an acknowledgement in response to receipt of that packet. The sending system tracks receipt of acknowledgements by intended recipient and retransmits any unacknowledged packets so as to positively assure the packets are received. Receiving systems send negative acknowledgements to request retransmission of missing packets. Negative acknowledgement suppression is implemented at both the sender and receiver to prevent a flood of negative acknowledgements from overwhelming the network. Packets are transmitted by the sending system at a transmission rate selected to avoid any adverse impact on the packet loss rate of the network.

EP 0 973 294 A2

ble solutions using efficient connectionless data transfer mechanisms, such as UDP multicast over IP networks. It would be a further advancement for such a method to further strongly couple response messages from each receiving system to the sending system in order to provide a bi-directional communication path.

SUMMARY AND OBJECTS OF THE INVENTION

[0009] It is an object of the present invention to reduce the amount of network traffic associated with reliably sending a small data message from a sending system to a number of receiving systems.

[0010] It is another object of the present invention to utilize negative suppression at both the sending system and at each receiving system to reduce network traffic.

[0011] For one aspect of the present invention, small messages are reliably sent on a statistically reliable basis so that the sending system is reasonably assured that all or almost all receiving systems have received the message while another aspect of the present invention positively assures that small messages were received by all the receiving systems.

[0012] Additional objects and advantages of the invention will be set forth in the description which follows, and in part will be obvious from the description, or may be learned by the practice of the invention. The objects and advantages of the invention may be realized and obtained by means of the instruments and combinations particularly pointed out in the appended claims.

[0013] To achieve the foregoing objects, and in accordance with the invention as embodied and broadly described herein, a method and computer program product for efficiently and reliably transmitting data messages from a sending system to a number of receiving systems is provided.

[0014] To overcome the problems in the prior art, two protocols have been developed. The base protocol, generally referred to as Statistically Reliable Transmission or statistical reliability mode, relies on a probabilistic model that can be tuned to reduce the probability that any single system did not receive a message to an arbitrarily small number thus essentially ensuring that all systems receive a message. For those situations that the statistical model is insufficient and receipt must be guaranteed, minor modifications can be made to the protocol to produce a Positive Reliability Transmission protocol or positive reliability mode where systems that do not receive a message can be identified and steps can be taken to ensure they receive the message. Decisions on which mode to use can be made on a per message basis by an application, or on a per-sender or per-site basis by systems management.

[0015] Both protocols are based on UDP and both protocols multicast UDP packets to one or many recipients. The basic protocol relies on the transmission of multiple packets. Thus, when a message fills less than a specified minimum number of packets, the message is expanded to fill the required minimum number of packets. The packets are numbered so that a recipient can determine if the entire message has been received. The packets are sent to the intended recipients using a pacing algorithm that regulates the speed at which packets are sent. The pacing algorithm recognizes that the packet transmission rate generally influences the packet loss rate in the network. Pacing the packets prevents the packet transmission rate from adversely influencing the packet loss rate.

[0016] When the positive reliability transmission mode is used, an ACK requested flag is set once every Nth packet. The collection of N packets is referred to as "ACK window" or "transmission window." Setting the ACK request flag signals the recipient to positively acknowledge receipt of that packet by sending an ACK to the sender. Furthermore the last packet in the transmission window has the ACK requested flag set. The ACK requested flag is not used in the statistically reliable transmission mode.

[0017] Since multiple packets are sent in a message, the probability that a system will receive at least one of the packets in a message is increased. By adjusting the minimum number of packets sent per message based on the packet loss rate of the system, the probability that a system will receive at least one packet can be reduced to a very small number. Systems that receive only part of a message can identify its incompleteness and send a NAK that triggers a retransmission.

[0018] In a large network, it will usually occur that many systems may not have received at least part of a message. If each system sent a NAK, then the flood of NAKs could overwhelm the network. The invention employs NAK suppression techniques on both the sender side and recipient side. The recipients calculate a delay time based on a defined algorithm that will be used to send a NAK to the sender. This reduces the number of NAKs received by the sender. In response to a NAK the sender will retransmit the missed packet. Any additional NAKs received by the sender for the same packet will be ignored for a predetermined period of time after retransmission of the packet. This further reduces the traffic on the network by giving the retransmitted packet time to be received by any system that may have missed it. In addition, each retransmit increases the probability that every system in the network will receive at least one packet in a message. The NAK/retransmit procedure is repeated for some period of time.

[0019] In the positive reliability mode, the sender listens for and tracks ACKs by recipient. Thus, any recipient that does not return an ACK can be identified. Periodically, all systems that have not returned an ACK for a particular transmit window are identified and the last packet of the transmit window is resent to them.

the general context of computer-executable instructions, such as program modules, being executed by a personal computer. Generally, program modules include routines, programs, objects, components, data structures, etc. that perform particular tasks or implement particular abstract data types. Moreover, those skilled in the art will appreciate that the invention may be practiced with other computer system configurations, including hand-held devices, multiprocessor systems, microprocessor-based or programmable consumer electronics, network PCs, minicomputers, mainframe computers, and the like. The invention may also be practiced in distributed computing environments where tasks are performed by remote processing devices that are linked through a communications network. In a distributed computing environment, program modules may be located in both local and remote memory storage devices.

[0026] With reference to Figure 1, an exemplary system for implementing the invention includes a general purpose computing device in the form of a conventional personal computer 20, including a processing unit 21, a system memory 22, and a system bus 23 that couples various system components including the system memory to the processing unit 21. The system bus 23 may be any of several types of bus structures including a memory bus or memory controller, a peripheral bus, and a local bus using any of a variety of bus architectures. the system memory includes read only memory (ROM) 24 and random access memory (RAM) 25. A basic input/output system 26 (BIOS), containing the basic routines that helps to transfer information between elements within the personal computer 20, such as during start-up, is stored in ROM 24. The personal computer 20 further includes a hard disk drive 27 for reading from and writing to a hard disk, not shown, a magnetic disk drive 28 for reading from or writing to a removable magnetic disk 29, and an optical disk drive 30 for reading from or writing to removable optical disk 31 such as a CD ROM or other optical media. The hard disk drive 27, magnetic disk drive 28, and optical disk drive 30 are connected to the system bus 23 by a hard disk drive interface 32, a magnetic disk drive-interface 33, and an optical drive interface 34, respectively. The drives and their associated computer-readable media provide nonvolatile storage of computer readable instructions, data structures, program modules and other data for the personal computer 20. Although the exemplary environment described herein employs a hard disk, a removable magnetic disk 29 and a removable optical disk 31, it should be appreciated by those skilled in the art that other types of computer readable media which can store data that is accessible by a computer, such as magnetic cassettes, flash memory cards, digital video disks, Bernoulli cartridges, random access memories (RAMs), read only memories (ROM), and the like, may also be used in the exemplary operating environment.

[0027] A number of program modules may be stored on the hard disk, magnetic disk 29, optical disk 31, ROM 24 or RAM 25, including an operating system 35, one or more application programs 36, other program modules 37, and program data 38. A user may enter commands and information into the personal computer 20 through input devices such as a keyboard 40 and pointing device 42. Other input devices (not shown) may include a microphone, joystick, game pad, satellite dish, scanner, or the like. These and other input devices are often connected to the processing unit 21 though a serial port interface 46 that is coupled to the system bus, but may be connected by other interfaces, such as a parallel port, game port or a universal serial bus (USB). A monitor 47 or other type of display device is also connected to the system bus 23 via an interface, such as a video adapter 48. In addition to the monitor, personal computers typically include other peripheral output devices (not shown), such as speakers and printers.

[0028] The personal computer 20 may operate in a networked environment using logical connections to one or more remote computers, such as a remote computer 49. The remote computer 49 may be another personal computer, a server, a router, a network PC, a peer device or other common network node, and typically includes many or all of the elements described above relative to the personal computer 20, although only a memory storage device 50 has been illustrated in Figure 1. The logical connections depicted in Figure 1 include a local area network (LAN) 51 and a wide area network (WAN) 52. Such networking environments are commonplace in offices' enterprise-wide computer networks, intranets and the Internet.

[0029] When used in a LAN networking environment, the personal computer 20 is connected to the local network 51 through a network or adapter 53. When used in a WAN networking environment, the personal computer 20 typically includes a modem 54 or other means for establishing communications over the wide area network 52, such as the Internet. The modem 54, which may be internal or external, is connected to the system bus 23 via the serial port interface 46. In a networked environment, program modules depicted relative to the personal computer 20, or portions thereof, may be stored in the remote memory storage device. It will be appreciated that the network connections shown are exemplary and other means of establishing a communications link between the computers may be used.

[0030] In certain environments, it is important to be able to transfer messages that fill relatively few packets from one computer to one or many other computers in a very reliable way. For example, Figure 2 illustrates a basic network connection where an operator console 70 is used to control several mid-tier servers 72 and end clients 76. Such an environment may be representative, for example, of a situation where multiple systems in a distributed network are managed from a central operator console. To perform such management, it is often necessary to have a highly reliable communications path between the operator console and each of the managed systems. Furthermore, management functions may require bi-directional communications where messages are sent from an operating console to multiple managed systems and the individual managed systems respond to the message. The messages required to perform such centralized management are, in general, fairly short consisting of one packet to several hundred or thousand pack-

$$p_k = \binom{n}{k} p^k (1-p)^{n-k}$$

where:

p_k is the probability of receiving k packets
n is the total number of packets transmitted
k is the number of packets received
p is the probability of receiving a packet.

[0039] The probability that none of the transmitted packets are received by a recipient is given by:

$$p_f = p_o = (1-p)^n$$

the probability of receiving all the packets transmitted is given by:

$$p_s = p_n = p^n$$

and the probability of receiving at least one packet but less than all the packets is given by:

$$p_{ps} = 1-(p_o+p_n) = 1-(p_f + p_s)$$

[0040] As an example, if there are one thousand recipients, three packets are transmitted, and the packet loss rate is twenty percent, then the probability of failure, the probability of success, and the probability of partial success is given by:

$$P_{ps} = 1-.2^3 = .008$$

$$P_{ps} = .8^3 = .512$$

$$P_{ps} = 1-.512 + .008 = .480$$

Thus, statistically 512 recipients can be expected to receive all three packets, 480 recipients will receive some of the packets, and 8 recipients will receive none of the packets. The table below presents the probability of failure as a function of the packet loss rate and the number of packets transmitted. As illustrated in the table, for moderate packet loss rates even a relatively few transmitted packets result in a very low probability of complete failure. Thus, by adjusting the number of packets transmitted for a particular packet loss rate, the probability of failure can be reduced to a very small number.

Table 1

Probability of Failure as a Function of the Packet Loss Rate and Number of Packets Transmitted											
Number of Packets Transmitted		1	2	3	4	5	6	7	8	9	10
Packet Loss Rate	10%	0.1	0.01	0.001	0.0001	1E-05	1E-06	1E-07	1E-08	1E-09	1E-10
Packet Loss Rate	20%	0.2	004	0.008	0.0016	0.0003	6E-05	1E-05	3E-06	5E-07	1E-07
Packet Loss Rate	30%	0.3	0.09	0.027	0.0081	0.0024	0.0007	0.0002	7E-05	2E-05	6E-06

Table 2 (continued)

Packet Type	ASCII Character	Used For	Packet Layout
Data	x'05'-ENQ	Transmit and Receive Data	2

[0044] Reserved field 90 is not used and is reserved for future enhancements. Bitmap field 92 contains flags that are used by the protocol for various purposes. Table 3 below identifies the flags contained in bitmap 92 and their purpose.

Table 3

Bit	Meaning	Values	Notes
x0001	EOM	0 or 1	Set in the last data packet for the message.
x0002	ACK Requested	0 or 1	Set in positive reliability mode when the sender wants an ACK in response.
x0004	Reply Requested	0 or 1	Set when the sender wants a response message.
x0008	Reply Requested Protocol	PosRel or StatRel	Required when Reply Requested is Specified. Default is PosRel
x0010	Reply Message	0 or 1	Set on all packets in a Response message
x0020	Reliability Mode	PosRel or StatRel	Reports the Reliability mode of all packets in a message.
Remainder	RESERVED	0	

[0045] Message number 94 is a field that contains a message ID number for the particular message. The message ID number is the same for all packets in a particular message. In addition, when replies are sent to a particular message, the reply also contains the message number of the message so that when the reply is received, the reply can be identified as being received in response to a particular message. Use of the message number allows requests and responses to be tightly bound so that the system can identify which responses go with which requests without enforcing any particular type of synchronous or connection-based protocol.

[0046] Packet sequence number 96 is a number that is incremented for each packet in the message. This allows a recipient to identify whether all packets of the message have been received in order to send NAKs requesting retransmission of missed packets. Response port 98 is the UDP port number that should be used for returning responses to the message. Data length field 100 identifies the length of data field 102 so that a recipient can extract all of the data field. Data field 102 contains the data to be transmitted in the packet.

[0047] As indicated in Figure 5, the packets have various layouts, illustrated as layout 0, layout 1, and layout 2. The packet layout for a particular packet type is summarized in Table 2.

[0048] Although some detail is presented in Figure 5 with respect to a particular packet implementation, such is presented by way of example only. Thus, Figure 5 should not be construed as limiting the scope of the present invention. Other embodiments may utilize different packet types, according to the particular feature set implemented by that particular embodiment. In general, however, most implementations will contain at least the destination ID, the sender ID, each of which comprise an IP address and a port number, and a data field for those packets that transfer data. The purposes of the other fields have been previously explained and, as illustrated more completely in the remainder of this discussion, are used to implement or enhance various feature sets that may be incorporated into various embodiments of the present invention.

[0049] Returning now to Figure 2, the conceptual operation of the statistical reliability mode of the present invention will be described. Consider that operator console 70 wishes to send a particular message to end clients 76. Operator console 70 will take the message and, as described in conjunction with Figure 4 above, will create at least the minimum number of packets necessary for the desired level of statistical reliability. The packets will then be transmitted using a multicast protocol, such as IP multicast, indicated in Figure 2 by IP multicast packets 74. The multicast packets will be received by mid-tier sewers 72 and evaluated. Since the packets are ultimately destined for end clients 76, mid-tier sewers 72 will transmit further multicast packets, such as IP multicast packets 78, to end clients 76.

[0050] Based on the statistical model above, some end users may receive all packets, some may receive some packets, and some may receive no packets. By examining the packet sequence number and, perhaps, the end of the message flag, systems that receive less than all the packets can transmit a NAK requesting retransmission of the missed packets. For example, if an end client 76 does not receive a packet in a message from mid-tier sewer 72, then end client

nologies, If sending system 104 is a multithreaded system, then sending system 104 may implement sender 108 and NAK receiver 118 as separate threads. Other appropriate implementations may also be utilized.

[0058] Referring now to Figure 7, a flowchart representing the processing steps taken by the sending system according to one embodiment of the present invention in order send a message to receiving systems is shown. Figure 7 may represent for example, the processing performed by sender 108 and NAK receiver 118 of Figure 6. After beginning at step 122, the message is divided into at least the requisite minimum number of transmission packets for desired reliability in step 124. This occurs as previously described in conjunction with Figure 4 above. Note that if the message doesn't require all of the packets then extra packets are added as previously described above to arrive at the minimum number of packets. These padding packets, even if not carrying data, are used to determine completeness of the message at the receiving system. The last packet will have the Last Packet or end of message flag set, and each packet will have a unique sequential packet number. Step 124 is, therefore, only one example of a means for dividing a message into a plurality of individual transmission packets and for adding additional transmission packets if the message fills less than a minimum number of individual transmission packets.

[0059] After the transmission packets have been created, the message is sent by sending each packet onto the IP network using the UDP multicast facility to a group of designated receiving systems. The transmission rate of the packets can influence the packet loss rate of the network. For example, in a network having routers and other systems that buffer and forward messages, if the packets are transmitted too fast, the buffer space can overflow, resulting in lost packets. Embodiments within the scope of the present invention may, therefore, comprise means for pacing transmission of packets. By way of example, and not limitation, in Figure 7 such means is illustrated by step 126 which sends packets using a pacing algorithm designed to transmit packets in such a manner that the transmission has little or no effect on the packet loss rate. The pacing algorithm reduces network overhead since it is responsive to general network packet error trends and sends packets out at greater intervals (lower packet transmission rate) when the network is performing at a lower level. Pacing algorithms may thus comprise means for selecting a transmission rate based on network performance levels. Examples of such means are presented hereafter.

[0060] Ideally, a pacing algorithm should allow the system to transmit packets as rapidly as possible without adversely effecting the packet loss rate of the network In general, the packet loss rate is a very complex function of many different parameters, at least some of which fluctuate over time. Some of the factors are related to hardware and the particular network infrastructure being utilized to communicate between systems. Other factors are influenced by the internal processing load of individual systems on the network. Because so many factors can influence the packet loss rate of the network, it would be very difficult to calculate a priori what the packet loss rate would be for any given particular network configuration. Thus, the present invention utilizes an approach which takes a measurement of the network packet loss rate and adjusts the transmission rate based on the packet loss rate in order to prevent the transmission rate from adversely effecting the system packet loss rate.

[0061] Many different methodologies can be developed to use the packet loss rate of the system to determine a transmission rate for the sender. Each of these is properly an example of means for selecting a transmission rate. Pacing algorithms that perform well, however, should have certain characteristics. In general, the transmission rate for a sender will be limited between some minimum and some maximum value. The minimum value is necessary so that the sender transmits packets at at least some minimum rate no matter what the network conditions are. If the sender did not transmit any packets, then network protocols would break down completely. Thus, even in the worst conditions the sender needs to transmit packets. Placing a maximum value on the transmission rate is often advisable to prevent the sender from totally overwhelming the network. The minimum and maximum transmission rate are often set based on testing or other methodologies that create acceptable minimum and maximum values.

[0062] When the packet loss rate is used to adjust the transmission rate, typical pacing algorithms rely on damping, filtering, weighting, and smoothing techniques to achieve performance levels that approximate ideal conditions. In general, the selected pacing methodology should respond to actual increases or decreases in packet loss rate while being relatively immune to small numbers of continuous errors and small bursts of errors so that they do not immediately trigger a reduction to minimum transmission rates.

[0063] In general, it is very difficult to directly measure the network packet loss rate. However, measurements exist which are indirect indicators of the packet loss rate. One such measurement is the number of NAKs received in response to prior transmissions. Since the statistical reliability mode of the present invention require systems that received at least one, but less than all of the packets in a message to NAK for packets they did not receive, the NAK rate is an indicator of the system packet loss rate. In using the NAK rate as an indicator of the packet loss rate, certain factors must be taken into account. For example, if the system employs NAK suppression in order to reduce the firestorm of NAKs that may be created when packets are lost in the network, the NAK rate may be artificially low for a given packet loss rate. It is possible, however, to factor in the effects of NAK suppression so that the pacing algorithm exhibits appropriate sensitivity to the measured NAK rate. For example, if the NAK suppression methodology filters out a relatively fixed quantity of NAKs, then each NAK measured may represent many such possible NAKs, most of which were never sent. If the NAK suppression algorithm has a time varying component, such as a NAK suppression algorithm

$$\Delta_r = \begin{cases} -\Delta_{NAK} * R_{\Delta B} & \text{if } \Delta_{NAK} > 0 \\ R_{\Delta B} & \text{otherwise} \end{cases}$$

[0069] Where:

Δ_r is the change in change in transmission rate;
$R_{\Delta B}$ is the base change in transmission rate; and
Δ_{NAK} is the amount the measured *NAK* rate is above the desired *NAK* rate and is calculated as illustrated below.

In the above equation, the amount that the measured *NAK* rate is above the desired *NAK* rate is calculated using the equation:

$$\Delta_{NAK} = \begin{cases} R_{NAK} - T_{NAK} & \text{if } R_{NAK} - T_{NAK} > T_{\Delta NAK} \\ 0 & \text{otherwise} \end{cases}$$

[0070] Where:

Δ_{NAK} is the amount that the measured *NAK* rate is above the desired *NAK* rate;
T_{NAK} is the desired *NAK* rate;
$T_{\Delta NAK}$ is a damping factor that represents the difference in the *NAK* rate versus the prior period that must be exceeded before the transmission rate will be reduced; and
R_{NAK} is the measured *NAK* rate.

Often it is desirable to deal with the *NAK* rates and other *NAK* quantities as *NAKs* per second per 1,000 systems in the network. If this is the case, then R_{NAK} in the prior equation may be calculated by:

[0071] Where:

$$R_{NAK} = \left(\frac{NAK}{population}\right) 1000$$

R_{NAK} is the *NAK* rate per 1,000 systems in the population per second;
NAK is the number *NAKs* received per second; and
population is the number of systems in the network that may transmit *NAKs*.

[0072] The performance and operation of the above system of equations can be optimized by changing the various constants and thresholds identified in the equations. In addition, certain parameters generally require selection of an initial value before the system of equations is iterated. In general, selection of the values of the various parameters and the initial starting values will be based on empirical data measured during test operations of the system. The desired values will be heavily dependent upon the exact installation and type of network encountered. In one embodiment the following values are used:

$T_{NAK} = 5$	$R_{min} = 250$
$T_{\Delta NAK} = 2$	$R_{max} = 2{,}000$

list 150 rather than a pointer to a separate message buffer. The exact implementation will depend upon various considerations such as the implementation details of the protocol stack used by receiver 148 to receive packets from network 112.

[0080] When a gap in the packet sequence is detected by the reception of a packet with a non-sequential packet sequence number, a NAK wait timer is started. At the expiration of the NAK wait time, a NAK is transmitted to sending system 144 to notify sending system 144 that a packet has been missed. Embodiments within the scope of this invention may comprise means for transmitting a request for packet retransmission. By way of example, and not limitation, in Figure 8 such means is illustrated by NAK timer 152. NAK timer 152 may watch message receive list 150 to detect non-sequential packet numbers. In the alternative, receiver 148 may notify NAK timer 152 when non-sequential packets have been received. As yet another alternative, the detecting of a non-sequential packet number, as well as starting the NAK wait timer may be incorporated into receiver 148. How this functionality is implemented will be based on a variety of design choices that need to be made for any particular implementation. In the embodiment illustrated in Figure 8, conceptually NAK timer 152 is responsible for setting the NAK wait timer when a non-sequential packet is detected and for transmitting a NAK to sending system 144 when the appropriate NAK timer has expired.

[0081] The use of a NAK wait timer to determine when to send NAKs provides several advantages. As previously explained above, if every system that did not receive a packet sent a NAK as soon as the missed packed was detected, the network may be completely choked with NAKs. Thus, it is desirable to space NAKs between systems so that the NAKs that are sent are distributed in an appropriate manner so that the transmission characteristics of the network are not adversely affected. Furthermore, since lower-level protocols, such as UDP may not guarantee reception of packets in numerical order, it is often desirable to wait a short period of time before sending a NAK for the packet to see if the packet will show up. Thus, embodiments within the scope of this invention may comprise means for suppressing requests for retransmission of packets. As previously explained, such a means may be one component of an overall means for reducing network traffic when recovering lost packets. By way of example, and not limitation, such a means for suppressing requests for retransmission of packets may comprise a NAK suppressor incorporated into NAK timer 152.

[0082] A NAK suppressor may comprise a wide variety of structures and methodologies to reduce the flood of NAKs that would occur if receiving systems immediately sent a NAK when a packet was received out of sequential order. In its most basic form, the NAK suppressor comprises a NAK wait timer that is set to a random value when a packet is received out of numerical order. Setting the timer to a random value randomizes the probability of transmitting a NAK and distributes the number of NAKs transmitted over a longer time period. As disclosed below, much more sophisticated NAK suppression methodologies are available, however.

[0083] Returning for a moment to Figure 2, when operator console 70 multicasts a plurality of packets to end clients 76, some systems will receive all transmitted packets, some systems will receive less than all of the packets transmitted, and some systems will receive none of the packets transmitted. Ideally, operator console 70 would receive only one NAK for each packet that was not received by any system in the network. For example, if various systems in the network did not receive two packets among the many that were transmitted, then ideally operator console 70 would only receive one NAK for each of the two packets that were not received. Although it is difficult to achieve this ideal in practice, NAK suppressors that have certain characteristics can approach this ideal.

[0084] In one embodiment of the present invention, the NAK suppressor utilizes a timer to delay transmission of NAKs. The value of the timer is set using a pseudo random number having a specific probability density function that creates relatively few NAKs early on and relatively more NAKs later. By tailoring the probability density function on the NAK timer, a situation can be created where among a population of recipients that did not receive a particular packet, a few systems will NAK relatively soon after the packet is missed. As long as the system receives these NAKs and responds with the appropriate packets, systems whose NAK timer was biased toward a later time will receive the packet before their NAK timer expires. If, however, the sending system does not receive the NAK and respond with a retransmission of the appropriate packet, then relatively more systems will NAK. By matching the probability density function to the particular packet lost characteristics of the network performance approaching the ideal can be achieved.

[0085] When constructing the probability density function of the NAK timers, several factors must be taken into account. These factors may be summarized as follows:

1. The probability density function should have a fairly sharp corner near the minimum desired value to provide a definitive cutoff point for the minimum timer value.
2. The low-end probability density function should be decreased as the network size is increased so that the overall number of systems that NAK early is relatively independent of the network size.
3. The probability distribution of the probability density function should make it highly probable that a few systems will NAK relatively early.
4. The probability density function should be relatively immune to the minimum and maximum NAK wait times so that the probability that a NAK will be sent by a particular system is relatively unchanged as a percentage when the

and minimum and maximum timer values, the scale factor S may be selected by the equation:

$$S = \frac{T_{max} - R_{max}}{\log_{10}(M)}$$

[0094] where:

S is the scale factor in the equation above;
R_{max} is the maximum random value calculated previously;
T_{max} is the maximum timer value;
M is the number of systems.

[0095] Since we want a timer value that is greater than the minimum allowable timer value, steps must be taken to ensure that the scaled timer value, T_s, falls within the desired range.
Thus, if T_s is less than the allowable minimum timer value, the allowable minimum timer value is added to T_s to bring it above the minimum. In other words:

$$T_m = \begin{cases} T_s & \text{if } T_s > T_{min} \\ T_s + T_{min} & \text{otherwise} \end{cases}$$

[0096] Where:

T_m is the timer value which is guaranteed to be above the minimum;
T_s is the scaled timer value calculated above;
T_{min} is the minimum allowable timer value.

[0097] It is desirable in the above equation to add T_{min} to the scaled timer value if it is less than the minimum rather than simply truncating the scaled timer value to the minimum. Truncating the scaled timer value to the minimum would result in an unacceptably high probability density function at the lower end of the timer values. Adding the minimum produces no such undesirable effects. Finally, the above calculations may have resulted in some values being above the maximum allowable timer value. These may be brought into the desirable range by simply limiting them to the maximum allowable timer value if they are greater than the maximum allowable timer value. This will increase the probability density function around the maximum allowable timer value. However, since it is desirable to bias the probability density function to larger values, this does not create a problem. Thus, the timer value selected, T, is given by:

$$T = \begin{cases} T_m & \text{if } T_m < T_{max} \\ T & \end{cases}$$

[0098] where:

T is the selected timer value
T_m is the timer value calculated in the previous equation above; and
T_{max} is the maximum allowable timer value.

[0099] Returning now to Figure 8, NAK timer 152 thus selects a NAK wait timer value when packets are missed as previously discussed and, upon expiration of the NAK wait timer value, sends NAK 154 to sending system 144 in order to request retransmission of the appropriate packet. In response to NAK 154, sending system 144 will retransmit the appropriate packet or packets as indicated by packet retransmission 156. It should be noted that NAK timer 152 may utilize a protocol stack, such as a protocol stack previously illustrated in Figure 3 to transmit NAK 154. It is also possible for NAK timer 152 to use the principles previously described in conjunction with the statistical reliability mode of trans-

viously explained, the NAK wait timer operates so as to stagger multiple NAKs for the same packet coming from different systems. This reduces network congestion and reduces the total number of NAKs. If the receive wait timer expires at step 168, this indicates that a packet is "overdue" and unexpectedly late. The NAK wait timer will be started in step 194 and if the packet isn't received by original transmission or retransmission before the expiration of the NAK wait timer a NAK will eventually be sent. The NAK wait timer value may be selected with an appropriate probability density function, as previously described. If the MSG life timer expires at step 170, then too much time has elapsed for receiving all of the packets to assemble the message thereby indicating that an error of some sort has occurred. There may some circumstances when a packet can never be delivered to the receiving system despite repeated sending of NAKs. The error indication is handled at step 174 before processing proceeds back to the top to await arrival of more packets or expiration of a timer. As previously mentioned, the error handling generally includes deleting any partially received message, but may also include such actions as sending notification of the failed reception to an appropriate error logging facility or other appropriate entity.

[0107] At step 178, the receiving system waits for packets as well as the expiration of timers (as represented by the dashed box 166). As packets are received at step 178, they are placed in a buffer, such as message receive list 150 of Figure 8, for reassembling the actual data message at step 180. If this packet is the first received packet for a new message (regardless of packet sequence number) as determined at decision block 182, the MSG life timer is started at step 184. Reception and placing of packets into an appropriate buffer may be performed by receiver 148 of Figure 8. The setting of MSG life timer in steps 182 and 184 may be performed either by receiver 148 or message life timer 158 of Figure 8, depending upon the exact implementation.

[0108] Since a packet has just been received, the receive wait timer is set/reset at step 186. As explained previously, the receive wait timer allows a time interval through which packets are expected to arrive from the sending system once a message has been started. This interval may be related to the pacing algorithm so that the two cooperate to avoid excessive NAKs that might occur when network performance is already degraded to the degree that the pacing algorithm is stretching times between packets that may exceed the receive wait timer interval value. This may also be performed either by receiver 148 or message life timer 158, depending upon the implementation. If the receipt of the packet has eliminated the need to send a NAK, the appropriate NAK wait timer is also cancelled at step 186.

[0109] At decision block 188, the receiving system determines if the packet is out of sequential order. If not, a determination is made at decision block 190 whether the message is completed. If the message is not completed, processing returns to waiting on timer expirations and the arrival of more packets. Otherwise, once the message is complete as determined at decision block 190 the completed message is passed up to the application at step 192 before processing proceeds back to the top to await arrival of more packets or expiration of a timer. The test for out of order packets at decision block 188 is performed by receiver 148 while the message complete test may be performed by either or both of receiver 148 or message handler 160. For example, when the message is complete receiver 148 can set the message complete flag which signals message handler 160 to pass the message to the application.

[0110] If a packet has been "skipped" as determined at decision block 188, it may be that the packet has been lost or dropped therefore requiring a NAK to be sent. For this reason the NAK wait timer is set at step 194 for the skipped packet in a fashion previously described.

[0111] Although the embodiment in Figure 9 uses separate receive wait timers and NAK wait timers, the effect of the two timers can be accomplished by using only NAK wait timers. This can be illustrated by examining the effect of the receive wait timers in Figure 9. The receive wait timers add a fixed wait time to the NAK wait timer in the situation when an expected packet does not arrive. The same effect can be achieved by setting a NAK wait timer whenever a packet is received with the NAK wait timer having a minimum value equal to the desired receive wait plus the regular minimum value of the NAK wait timer. If a packet is received out of order, the NAK wait timer is still set with the regular minimum value of the NAK wait timer.

[0112] In certain situations, the reliability provided by the statistical reliability mode of the present invention is insufficient and absolute guaranteed reliability is required. In such a situation, the present invention can operate in a positive reliability mode. The positive reliability mode adds additional processing steps and functionality to the basic statistical reliability mode in order to guarantee reception by each intended recipient of the message that is capable of receiving message packets. The positive reliability mode is able to identify any intended recipients that did not receive the message so that a rebroadcast to those intended recipients may be achieved. The following discussion focuses on the positive reliability mode of the present invention. It should be noted that embodiments within the scope of this invention may switch between positive reliability mode and statistical reliability mode on a message-by-message basis by an application, or on a per-sender or per-site basis by systems management. Thus, in embodiments where it is desirable to provide both the statistical reliability mode and the positive reliability mode incorporation of the additional functionality for the positive reliability mode is needed.

[0113] Transmission in the positive reliability mode proceeds largely as previously described in conjunction with the statistical reliability mode, with some important additions. Basically, the sending system builds a transmission list (TList) of all the intended recipients. The message to be transmitted is then broken into packets for transmission, with addi-

in one embodiment send verifier 202 begins scanning transmission list 198 after a sufficient time has elapsed that an ACK is expected to be returned. This time delay is generally set based on the expected propagation delay in the network, the time that it takes to process received packets at the receiving system and the propagation delay for the ACK back to the sending system. In addition, since the ACK request flag is only set once each transmit window, it may be desirable to add at least one transmit window's worth of delay onto the expected value. Furthermore, it may be desirable to account for the current pacing output since it influences the transmission rate of the packets. Once send verifier thread scans transmission list 198, it should periodically rescan the list. In one embodiment transmission list 198 is scanned with the frequency of one-half times the expected ACK delay time.

[0121] Since the information in transmission list 198 is only required until receipt of message is assured, embodiments within the scope of this invention may comprise means for freeing memory no longer needed by transmission list 198. In Figure 10, such means is illustrated, by way of example, by memory return 204. When a transmit window has been completely acknowledged by all recipients, the memory used for both that portion of the message buffer and the entry in the transmit list for that transmit window can be freed. In other words, when the corresponding ACK bits are set in the transmit window bitmap for all IP addresses that are identified to receive the message, the memory containing the transmit window may be freed and the pointer to the message buffer may be adjusted. Once the message has been completely received by all intended recipients, the transmit list may also be discarded. Various embodiments of the present invention may utilize different policies for initiating scans by memory return 204. In one embodiment memory return 204 runs on the same schedule as send verifier 202. In other embodiments different schedules may be utilized.

[0122] Referring now to Figures 11A and 11B, a flowchart representing the processing steps taken by the sending system according to the present invention in order to send a message to a receiving system is shown. After beginning at step 206 of Figure 11A, the sending system will create a transmit list or TList at step 208 so that the ACK status for all receiving systems may be tracked. As previously described, one tracking list is created for each transmitted message. The tracking list may include such information as the message number, a pointer to the message buffer, and for each recipient, the recipient's UDP address, a transmit list reply bit, a transmit window bitmap, and the time stamp of the latest ACK or NAK received.

[0123] At step 210, the message is divided into the requisite minimum number of transmission packets for desired reliability. This proceeds as previously explained, with extra packets added to arrive at the minimum number of packets if required. These padding packets, which may not carry necessary data, are used to determine completeness of the message at the receiving system. The last packet will have the Last Packet flag set, and each packet will have a unique sequential packet number.

[0124] With the packets created, the sending system will set the ACK Request flag for every N^{th} packet. The N interval of packets is chosen based on desired positive reliability with smaller numbers creating greater network traffic but providing higher levels of reliability. Thus, if the ACK request flag is set every N^{th} packet a transmit window is N packets long.

[0125] The message is sent by sending each packet onto the IP network using the UDP multicast facility at step 213 to the designated receiving systems. The packets will not be sent out all together or as rapidly as possible in the illustrative embodiment. They may be sent out according to a pacing algorithm, such as that previously described. As explained previously, the pacing algorithm reduces network overhead since it is responsive to general network packet error trends and sends packets out at greater intervals when the network is performing at a lower level. Steps 208, 210, 212, and 213 can be performed by sender 108 of Figure 10.

[0126] At step 214, the sending system will wait for NAKs from the receiving systems so that it can retransmit any missed packets as well as the ACKs from each receiving system for each N^{th} packet received. Listening for NAKs and retransmitting any required packets may be performed by NAK receiver 118, while listening for ACKs may be performed by ACK receiver 200. In order to reduce the amount of network traffic caused by retransmission, embodiments within the scope of this invention may comprise means for suppressing packet retransmission. By way of example, and not limitation, in Figure 11A such means is represented by retransmission suppressor 215. Retransmission suppressor 215 reduces duplicate retransmission of the same packet when two NAKs for the same packet are received within a designated time period. At decision block 216, a test is made to see if there is a suppression timer running for this packet. If so, the NAK is ignored since a retransmission has already occurred and that particular receiving system may already or may shortly have the missing packet. This prevents extra retransmissions of the same packet that will tend to degradate network performance.

[0127] If the suppression timer is not running as determined at decision block 216, one is started for this packet at step 218 and the packet is retransmitted, again using UDP multicast, at step 220. Multicast is used so that all receiving systems that may have missed the packet previously will have an opportunity to receive it. As previously described, in conjunction with NAK suppressor 130 of Figure 7, NAK suppressor 215 may work in conjunction with receiving system processing that reduces the actual number of NAKs generated. Decision block 216, step 218, and step 220 may be implemented by NAK receiver 215 of Figure 10.

[0128] If an ACK is received at step 214, the sending system will mark the corresponding receiving system in the TList

the packet loss rate of the network.

4. A method as recited in claim 1 further comprising the step of identifying, by the sending system in response to a negative acknowledgment received from a receiving system, whether the unreceived packet has already been retransmitted within a designated time period and if so, then ignoring said negative acknowledgment, otherwise retransmitting the unreceived packet.

5. A method as recited in claim 1 further comprising the steps of:

setting a negative acknowledgement wait timer value at each receiving system which determines that a transmission packet making up the message has not been received, said timer value being selected between a minimum negative acknowledgement wait time and a maximum negative acknowledgement wait time according to a given probability density function; and
upon expiration of the negative acknowledgement wait timer value, if the missing transmission packet has not been received, then transmitting a negative acknowledgement to the sending system.

6. A method as recited in claim 1 further comprising each receiving system starting a receive message timer after each packet is received with the expectation that another packet will arrive before said receive message timer expires.

7. A method as recited in claim 1 wherein the sending system sends the transmission packets multicast out to the receiving systems within a group and each receiving system sends any applicable communication unicast back to the sending system.

8. A method as recited in claim 1 wherein each transmission packet further contains a reply request flag that, when set, indicates that each receiving system will send a reply message back to the sending system.

9. In a network comprising a sending system networked together with at least one receiving system, a method for efficiently and reliably transmitting a data message from the sending system to the at least one receiving system in a manner that minimizes network traffic while maintaining high reliability, the method comprising the steps of:

dividing the message to be sent into a plurality of data blocks each of which is carried by a sequentially identified transmission packet, and if the total number of transmission packets is less than a defined minimum number, then creating additional sequentially identified transmission packets until said defined minimum number are available;
transmitting said sequentially identified transmission packets from the sending system to the receiving systems;
waiting for a negative acknowledgment to be received from the at least one receiving system requesting retransmission of at least one transmission packet; and
upon receipt of said negative acknowledgement, determining if the requested at least one transmission packet has been transmitted within a designated period of time, and if so, then ignoring said negative acknowledgment, otherwise retransmitting the requested at least one transmission packet.

10. A method for transmitting a data message as recited in claim 9 further comprising the step of sending the packets using a pacing method that sends the transmission packets at a given transmission rate so that the transmission of the packets has little or no effect on the packet loss rate of the network.

11. A method for transmitting a data message as recited in claim 10 further comprising the step of setting an acknowledgment flag every N^{th} transmission packet.

12. A method for transmitting a data message as recited in claim 11 further comprising the step of creating a transmission list comprising the intended receiving systems that are to receive the message and a transmit window bitmap that is adapted to record receipt of acknowledgements received from each of the intended receiving systems for each transmission packet having an acknowledgment flag set.

13. A method for transmitting a data message as recited in claim 12 further comprising the steps of

waiting for an acknowledgment to be returned from a receiving system; and

means for receiving requests for packet retransmission and for retransmitting the requested packets; and

means for suppressing packet retransmission if a request for retransmission is received within a designated time after a given packet has been transmitted.

19. A computer readable medium as recited in claim 18 further comprising means for receiving acknowledgments from the receiving systems.

20. A computer readable medium as recited in claim 18 further comprising means for tracking receipt of messages by intended recipient so that receipt of individual packets by a given intended recipient can be confirmed.

21. A computer readable medium as recited in claim 18 further comprising means for identifying recipients which have not acknowledged receipt of packets.

22. A computer readable medium as recited in claim 18 further comprising means for retransmitting packets that have already been sent.

23. A computer readable medium as recited in claim 18 further comprising means for pacing transmission of packets so that the transmission of packets has little or no effect on the packet loss rate of the network.

24. A computer readable medium having computer executable instructions carried thereon for use in a network comprising a sending system networked together with a plurality of receiving systems, where messages are sent from said sending system to at least one receiving system using transmission packets designed to carry a designated quantity of data among a plurality of systems connected together in a network configuration, the computer executable instructions comprising:

means for receiving individual packets of a message comprised of a guaranteed minimum number of individual transmission packets;

means for storing received packets until the complete message can be reassembled in the proper order or until it is determined that the complete message will not be received;

means for transmitting a request for packet retransmission if a packet in the message has not been received; and

means for suppressing requests for retransmission of packets so that requests for retransmission from a plurality of receiving systems are not sent at the same time, said means for suppressing requests comprising means for selecting a wait time according to a defined probability density function.

25. A computer readable medium as recited in claim 24 further comprising means for removing an incomplete message when it is determined that the message will not likely be received in its entirety.

26. A computer readable medium as recited in claim 24 wherein the defined probability density function is constructed so that relatively few systems select shorter wait times and relatively more systems select longer wait times.

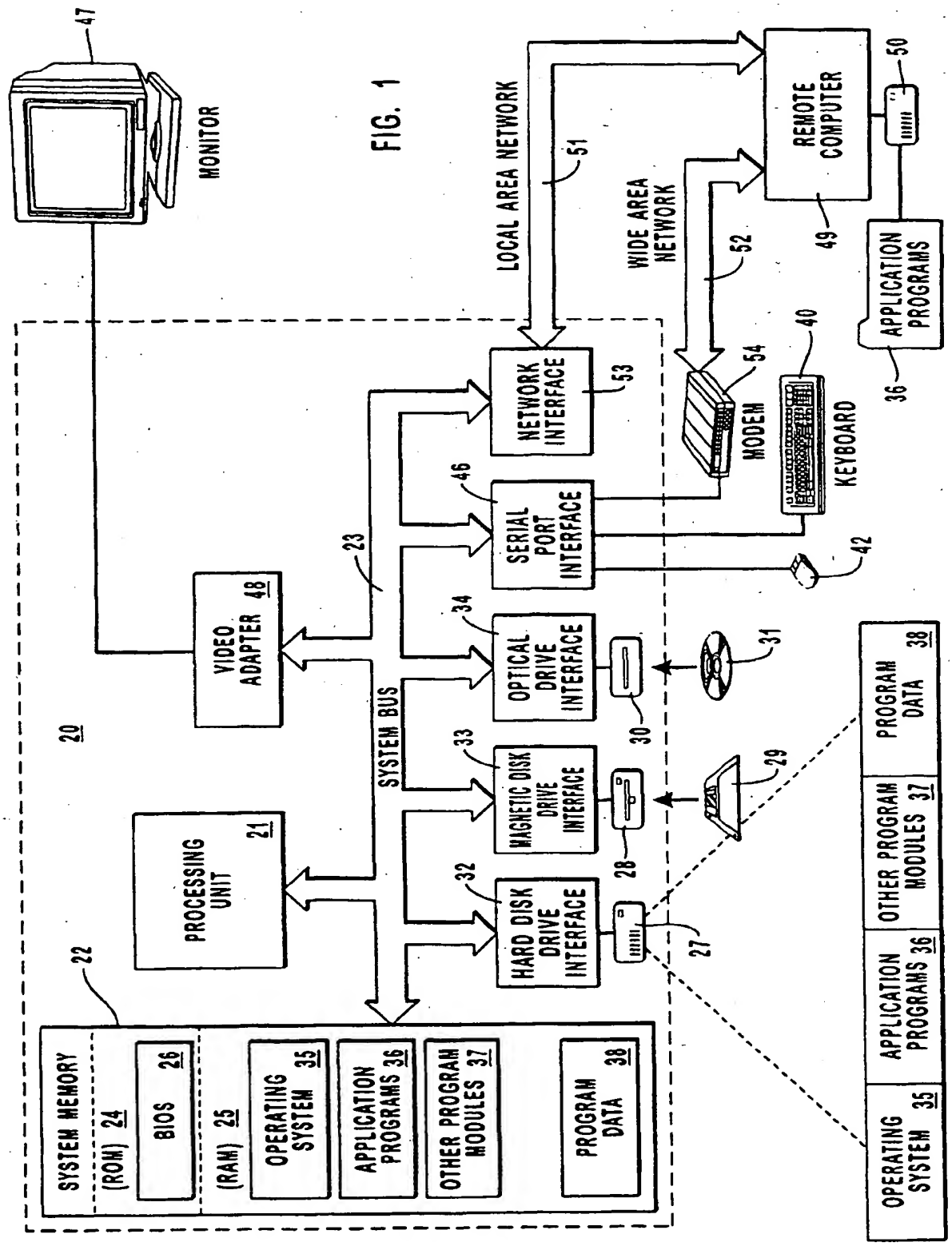
27. A computer readable medium as recited in claim 24 further comprising means for transferring a completely received message to the appropriate destination.

28. A computer readable medium as recited in claim 24 further comprising means for transmitting an acknowledgement when a packet is received that has an acknowledgment request flag set.

29. A computer readable medium having computer executable instructions carried thereon for use in a network comprising a sending system networked together with a plurality of receiving systems, where messages are sent from said sending system to at least one receiving system using transmission packets designed to carry a designated quantity of data among a plurality of systems connected together in a network configuration, the computer readable medium comprising:

a first computer readable medium having computer executable instructions for use in the sending system, said computer executable instructions comprising:

a sender comprising 1) a software component adapted to divide a message into a plurality of individual transmission packets and to add additional transmission packets if the message fills less than a minimum


47
MONITOR
FIG. 1
LOCAL AREA NETWORK
51
WIDE AREA NETWORK
52
REMOTE COMPUTER
49
50
APPLICATION PROGRAMS
36
40
KEYBOARD
54
MODEM
NETWORK INTERFACE
53
46
SERIAL PORT INTERFACE
42
23
VIDEO ADAPTER 48
34
OPTICAL DRIVE INTERFACE
31
30
SYSTEM BUS
33
MAGNETIC DISK DRIVE INTERFACE
29
28
20
PROCESSING UNIT
21
32
HARD DISK DRIVE INTERFACE
27
22
SYSTEM MEMORY
(ROM) 24
BIOS 26
(RAM) 25
OPERATING SYSTEM 35
APPLICATION PROGRAMS 36
OTHER PROGRAM MODULES 37
PROGRAM DATA 38
PROGRAM DATA 38
OTHER PROGRAM MODULES 37
APPLICATION PROGRAMS 36
OPERATING SYSTEM 35

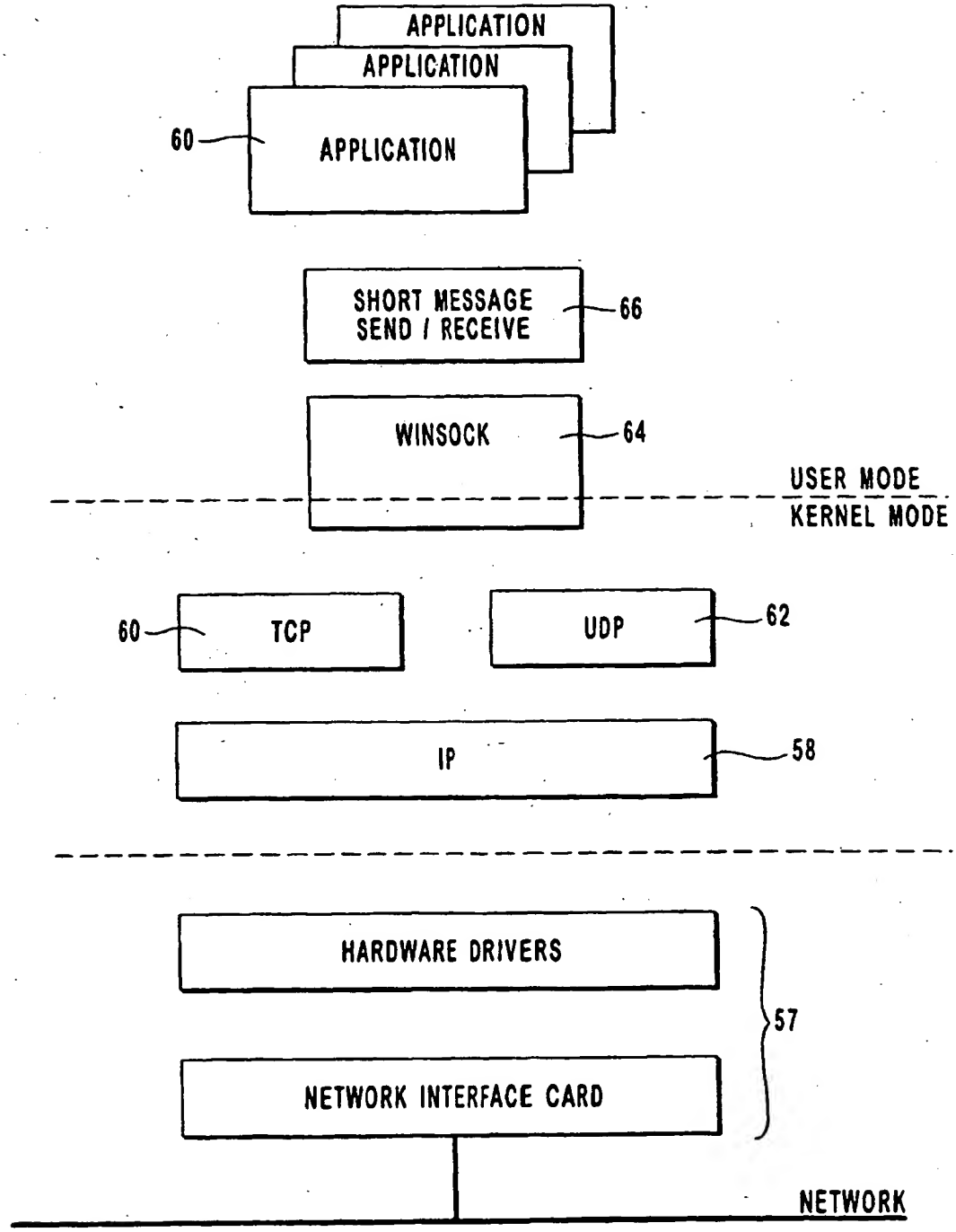

APPLICATION
APPLICATION
60
APPLICATION
SHORT MESSAGE
SEND / RECEIVE
66
WINSOCK
64
USER MODE
KERNEL MODE
60
TCP
UDP
62
IP
58
HARDWARE DRIVERS
57
NETWORK INTERFACE CARD
NETWORK

FIG. 3

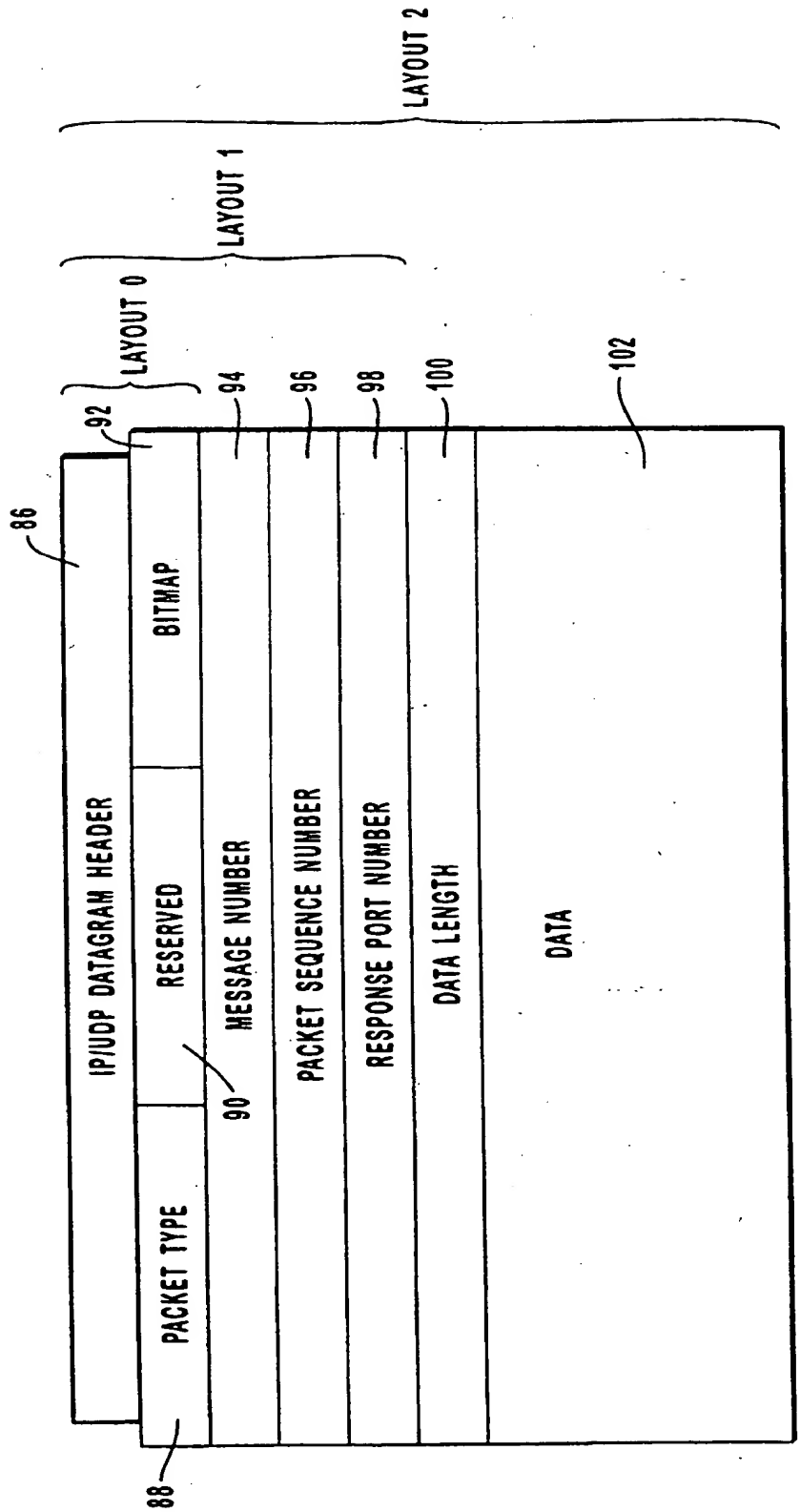

LAYOUT 2
LAYOUT 1
LAYOUT 0
86
92
94
96
98
100
102
IP/UDP DATAGRAM HEADER
BITMAP
RESERVED
MESSAGE NUMBER
PACKET SEQUENCE NUMBER
RESPONSE PORT NUMBER
DATA LENGTH
DATA
PACKET TYPE
90
88

FIG. 5

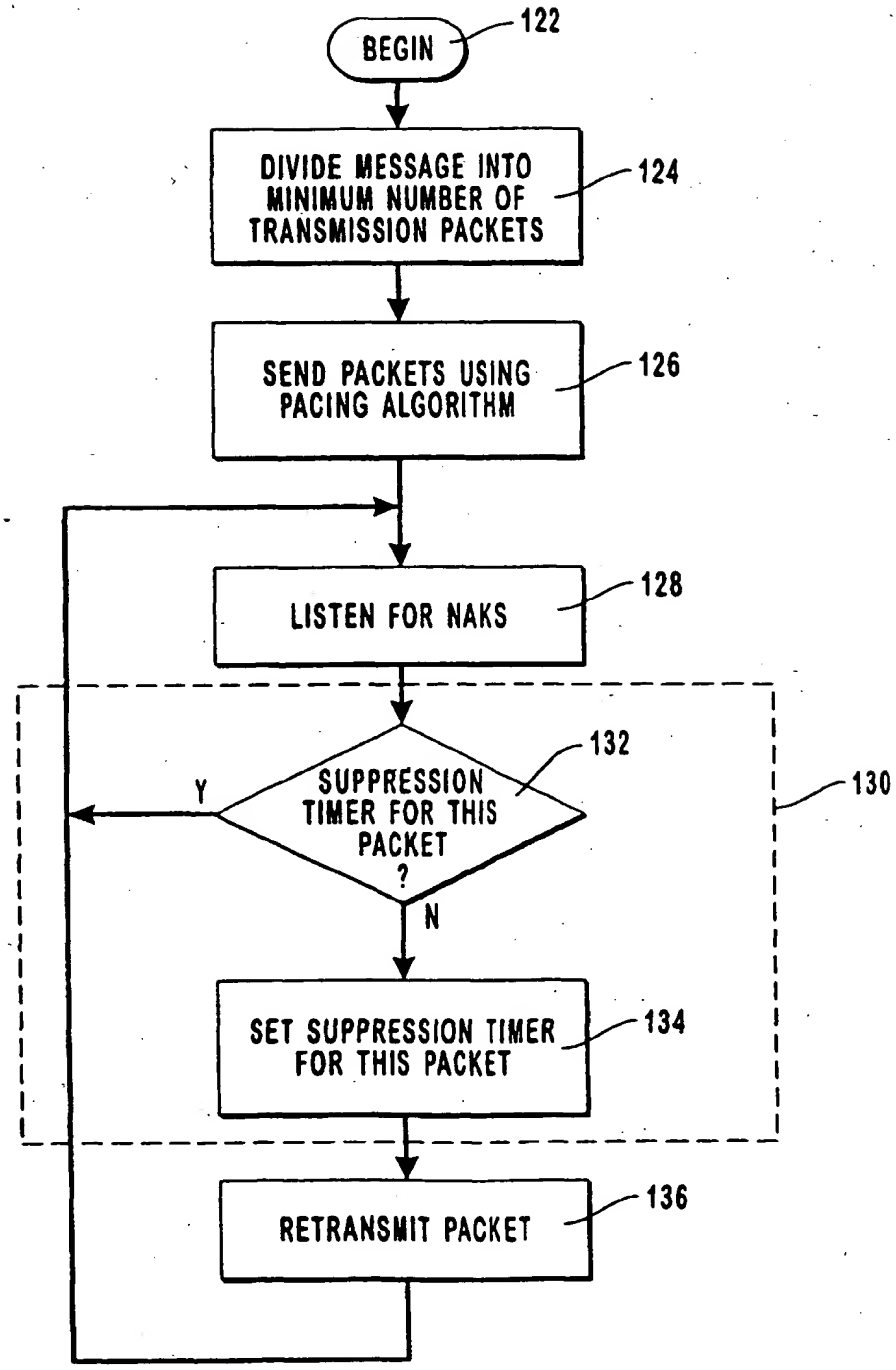

BEGIN
122
DIVIDE MESSAGE INTO MINIMUM NUMBER OF TRANSMISSION PACKETS
124
SEND PACKETS USING PACING ALGORITHM
126
LISTEN FOR NAKS
128
SUPPRESSION TIMER FOR THIS PACKET ?
132
Y
N
130
SET SUPPRESSION TIMER FOR THIS PACKET
134
RETRANSMIT PACKET
136

FIG. 7

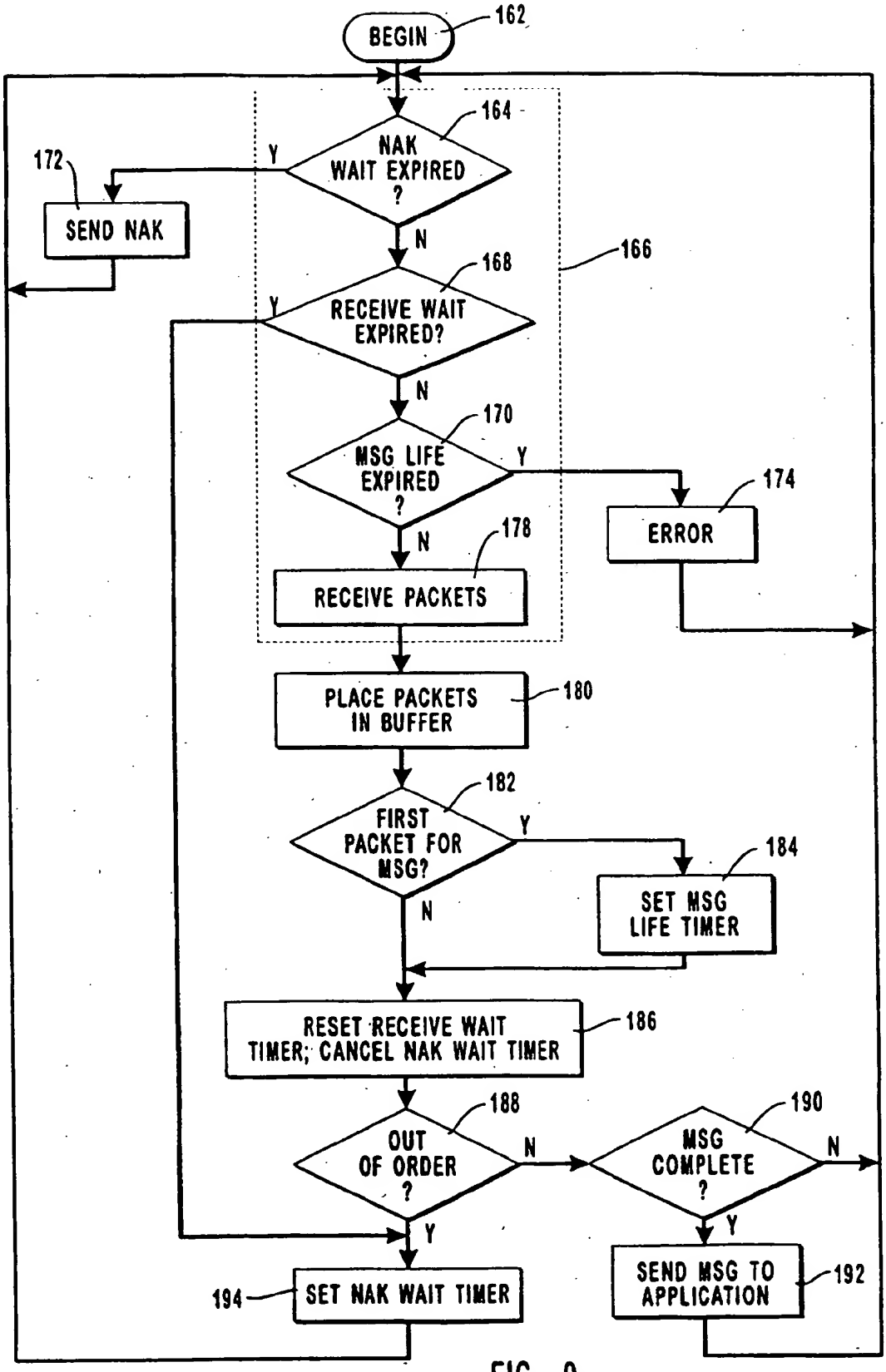

BEGIN
162
164
NAK WAIT EXPIRED ?
Y
172
SEND NAK
N
166
168
RECEIVE WAIT EXPIRED?
Y
N
170
MSG LIFE EXPIRED ?
Y
174
ERROR
N
178
RECEIVE PACKETS
PLACE PACKETS IN BUFFER
180
182
FIRST PACKET FOR MSG?
Y
184
SET MSG LIFE TIMER
N
RESET RECEIVE WAIT TIMER; CANCEL NAK WAIT TIMER
186
188
OUT OF ORDER ?
N
190
MSG COMPLETE ?
N
Y
Y
194
SET NAK WAIT TIMER
SEND MSG TO APPLICATION
192

FIG. 9

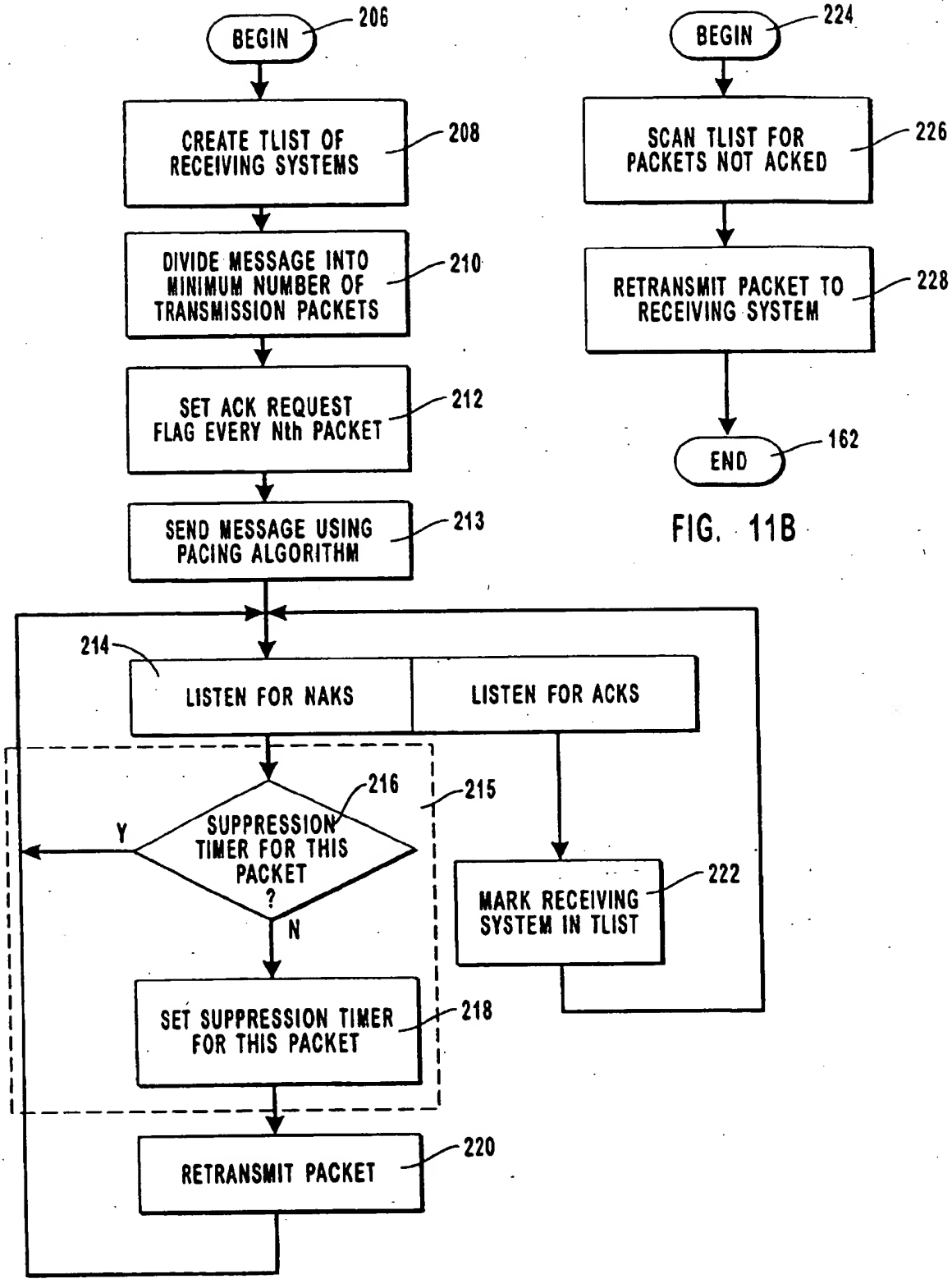


FIG. 11A

FIG. 11B

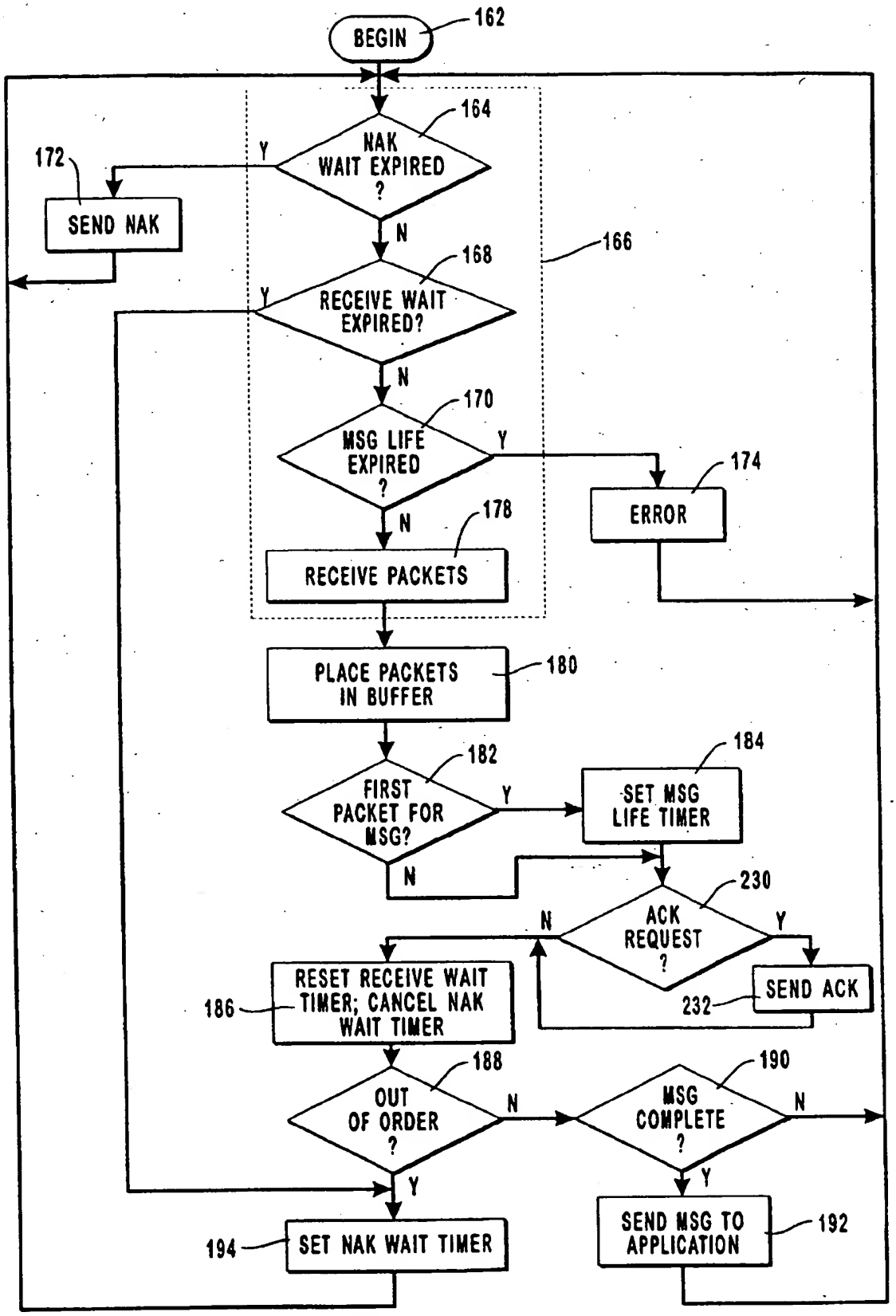

BEGIN
162
NAK WAIT EXPIRED ?
164
Y
N
SEND NAK
172
RECEIVE WAIT EXPIRED?
168
166
MSG LIFE EXPIRED ?
170
ERROR
174
RECEIVE PACKETS
178
PLACE PACKETS IN BUFFER
180
FIRST PACKET FOR MSG?
182
SET MSG LIFE TIMER
184
ACK REQUEST ?
230
SEND ACK
232
RESET RECEIVE WAIT TIMER; CANCEL NAK WAIT TIMER
186
OUT OF ORDER ?
188
MSG COMPLETE ?
190
SET NAK WAIT TIMER
194
SEND MSG TO APPLICATION
192

FIG. 13